Gitolite (git administration)

Introduction

Gitolite is an access control layer on top of git. Here are the features that most people see:

- Use a single unix user ("real" user) on the server.
- Provide access to many gitolite users:
  - they are not "real" users,
  - they do not get shell access.
- Control access to many git repositories:
  - read access controlled at the repo level,
  - write access controlled at the branch/tag/file/directory level, including who can rewind, create, and delete branches/tags.
- Can be installed without root access, assuming git and perl are already installed.
- Authentication is most commonly done using sshd, but you can also use http if you prefer (this may require root access).

Pre-requisites

Server

- Any Unix system with a POSIX compatible "sh" and a **sane** file system.
- Git version 1.6.6 or later. (Git 1.7.8 or later if you want to run the test suite).
- Perl 5.8.8 or later.
- Openssh (almost any version). Optional if you're using smart http.
- A dedicated Unix user for the hosting user, usually "git" but it can be any user, even your own normal one. (If you're using an RPM/DEB the install probably created one called "gitolite").

Also see the WARNINGS page for more on what gitolite expects on the server side.

TODO: which packages to install for Ubuntu 12.04 and 14.04 and Debian 6, 7 and 8?

Client

The gitolite client is an ordinary git client, documented at Git user manual

Installation

Server

Gitolite has only one server side "command" now, much like git itself. This command is `gitolite`. You don't need to place it anywhere special; worst case you run it with the full path.

"Installation" consists of the following options:

1. Keep the sources anywhere and use the full path to run the `gitolite` command.
2. Keep the sources anywhere and symlink *just* the `gitolite` program to some directory on your `$PATH`.
3. Copy the sources somewhere and use that path to run the `gitolite` command.

Option 2 is the best for general use.

There is a program called 'install' that helps you do these easily. Assuming your cloned the repo like this:

```
git clone git://github.com/sitaramc/gitolite
```

you can run the 'install' command in 3 different ways:

```
# option 1 gitolite/install # option 2 gitolite/install -ln # defaults to $HOME/bin (which is assumed to
exist) # ** or ** # or use a specific directory (please supply full path): gitolite/install -ln /usr/local
/bin # option 3 # (again, please supply a full path) gitolite/install -to /usr/local/gitolite/bin
```

Creating a symlink doesn't need a separate program but 'install' also runs `git describe` to create a VERSION file, which, trust me, is important!

Installing the software gets you ready to use it, but the first "use" of it is always the "setup" command.

The first time you run it, you need to have a public key file (usually from the admin's workstation) ready. If the main gitolite admin's username is "alice", this file should be named "alice.pub". Then, as the hosting user, run:

```
gitolite setup -pk alice.pub
```

If that command completes without any warnings, you should be done. If it had a warning, you probably supplied a key which already has shell access to the server. That won't work.

> *Normally, gitolite is hosted on a user that no one accesses directly -- you log on to the server using some other userid, and then `su - git`. In this scenario, there is no key being used for shell access, so there is no conflict.*
>
> *An alternative method is to use two different keys, and a host alias to distinguish the two.*
>
> *common errors has some links to background information on this issue.*

The 'setup' command has other uses, so you will be running it at other times after the install as well:

- To setup the update hook when you move existing repos to gitolite. This also applies if someone has been fiddling with the hooks on some repos and you want to put them all right quickly.
- To replace a lost admin key.
- To setup gitolite for http mode (run 'gitolite setup -h' for more info).

When in doubt, run 'gitolite setup' anyway; it doesn't do any harm, though it may take a minute or so if you have more than a few thousand repos!

Client

The gitolite client is an ordinary git client, documented at Git user manual

Configuration and administration

Gitolite

to administer gitolite:

```
git clone git:gitolite-admin
```

The directory tree is self explanatory

To update commit:

```
git commit -a #tell git about the changes you want to incorporate
git push # send it to the server
```

Set the "remote" of an "orphan", local git repo, to a new repo created with gitolite

This is a usual use case: a repo has been created on a development machine, before the repo created using gitolite on the server.

Assume you have an existing project you want to add in your new repository.

```
cd foo
git init
git add --all
git commit -m "Initial commit"
```

Link your local repository with your remote repository then push the local content to your remote repository.

```
git remote add origin git-bluelight:foo
git push --set-upstream origin master
```

References

TBC